# Energy Efficient, Context-Aware Cache Coding for Mobile Information-Centric Networks

Joshua Joy    Yu-Ting Yu
Mario Gerla
Network Research Laboratory
UCLA
{jjoy,yutingyu,gerla}@cs.ucla.edu

Ashish Gehani    Hasnain Lakhani
Minyoung Kim
Computer Science Laboratory
SRI International
firstname.lastname@sri.com

## ABSTRACT

In a mobile, intermittently connected information-centric network (ICN), users download files either from the original source or from caches assembled during previous downloads. Network coding has helped to increase download robustness and overcome "missing coupon" delays. Prior work has also shown that network coding depletes energy resources much faster than no coding. Our contribution here is to make coding more efficient, and to detect when it is not necessary, in order to prolong the life of mobile handhelds.

In the network coding context, Cache Coding (i.e., coding performed only on fully cached files) can prevent pollution attacks without significantly reducing diversity and performance with respect to unrestricted code mixing. Cache Coding introduces the first important means to reduce energy consumption by avoiding the extremely processor-intensive homomorphic code used in conventional unrestricted mixing networks. Our second contribution is to detect when Cache Coding is not required and disable it to save precious energy. The proposed Context-Aware Cache Coding (CACC) toggles between using Cache Coding and no coding based on the current network context (e.g., mobility, error rates, file size, etc). Our CACC implementation on Android devices demonstrates that the new scheme improves upon network coding's file delivery rate while keeping energy consumption in check.

*DEBS '16, June 20-24, 2016, Irvine, CA, USA*
© 2016 ACM. ISBN 978-1-4503-4021-2/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2933267.2940322

## CCS Concepts

•**Networks → Error detection and error correction;**

## Keywords

Network coding, Cache Coding, Pollution protection, ICN, Energy efficiency

## 1. INTRODUCTION

Tactical and emergency response scenarios require efficient, robust, and secure network communication among team members in order to quickly deliver data for situational awareness applications. Handhelds that opportunistically communicate with limited global knowledge have a dynamic and resource-constrained nature that demands efficient use of scarce available bandwidth. In such settings, networks of smart phones are formed in an ad-hoc and information-centric manner. This means that network operations are defined at the granularity of entire pieces of content, and that participating nodes may cache content on behalf of others. However, despite the remarkable increase in sensing, storage, processing, and communication capabilities in mobile devices today, efficient dissemination and storage of content at the volatile network edge remains a challenging research problem.

Mobile Ad Hoc Information Centric Networks (which we refer to as MANET ICN) have intermittent connectivity. This introduces the following challenges during file transfer and dissemination:

- *End-to-end connectivity not guaranteed*: A continuous path from source to destination may sometimes not occur. Nodes must therefore maintain coded caches of partial transfers and wait for contact with other nodes to continue transmission.

- *Partial caches*: Various caches contain different pieces of a file. A receiver does not know which caches contain which pieces of the file in advance and must ask each individual cache. It is important to parameterize the coding parameters to maximize throughput and minimize energy consumption.

- *Energy efficiency*: The tactical networks we are addressing here are largely comprised of handheld devices with limited battery life. Improving the resilience of the network requires an explicit tradeoff between energy consumption and transmission reliability. One of the key players in this tradeoff is network coding, as it improves reliability at the cost of extra energy.

Network coding [13] has been used in mobile ad hoc networks [19] for data dissemination to overcome the problem of intermittent connectivity. With Cache Coding [16], the content originator encodes the file being transferred. Intermediate nodes can re-encode only if they have cached the file. More precisely, each file consists of file fragments. The originator transforms the file into coded blocks, which are linear combinations of fragments of the file. These coded blocks are then propagated in the network. Note that in this scheme, a node can code only if it has possession of the entire file in its cache. Such nodes are either the originator or an intermediate node that happens to have cached the whole file. Thus, both originators and intermediate caches perform Cache Coding.

One major advantage of this approach is that the ordering of partially reassembled files is no longer needed for file transfer consistency. The requestor can retrieve arbitrary coded blocks from any node, and reassemble the original file from a sufficiently large number of linearly independent coded blocks without worrying about sorting them.

Unrestricted network coding, in which intermediate nodes perform unrestricted mixing (as opposed to mixing of fragments from the cached file), offers more diversity than Cache Coding. The restriction to Cache Coding thus accepts a greater cost in performance. However, as mentioned earlier, unrestricted coding is vulnerable to devastating pollution attacks launched by malicious intermediate nodes, and can corrupt all downstream blocks. Conventional solutions use homomorphic signatures to overcome these pollution attacks. Unfortunately, homomorphic codes are typically 100 times more processor-intensive than regular networking codes, and have a 100-fold increase in expenditure [18]. Cache Coding provides full protection against pollution attacks while avoiding the extra processing overhead of homomorphic coding [16]. This comes at the cost of a modest loss in code diversity and thus throughput efficiency, which is offset by enormous processing and energy savings.

Cache Coding, like network coding, improves file delivery in real-world scenarios with severely disrupted networks. The benefits are summarized here:

- *Overcoming intermittent connectivity*: Since end-to-end connectivity is not guaranteed, blocks are cached at intermediate nodes. A requestor can ask nearby nodes for network coded blocks. The neighbors pull coded blocks from their cache and, depending on context, either transmit them as they are or mix them and transmit new coded blocks.

- *Exploiting partial caches*: Nodes cache partial files that are encoded in the form of linearly independent "innovative" blocks. Since each coded block is as useful for decoding as any another, a requestor need not contact a particular cache to find missing blocks. Any node that has blocks for that file will do.

- *Leveraging alternative peer caches*: When a requestor discovers that the nearest cache is offline or is too busy to answer requests, it can ask other nearby coded caches for blocks since each network coded block is as useful as any other for decoding the file.

- *Efficient battery usage*: It has previously been shown that network coding achieves the minimum energy per bit required for reliable dissemination of files in environments with intermittent connectivity [29, 21]. However, network coding leads to shorter battery lifetimes due to the processing overhead [17, 11]. Our contribution, Context-Aware Cache Coding specifically addresses this problem. It detects when Cache Coding is required based on the context of the transfer. It then forwards packets as they come in, without extra code processing overhead, thus saving precious energy. This allows nodes to limit the use of Cache Coding (and corresponding energy consumption) to the situations when it is needed. At all other times, Cache Coding is automatically disabled.

The contribution of this paper is two-fold. First, we demonstrate that Cache Coding must be used to ensure reliable file delivery in situations where intermittent connectivity losses and/or severe network disruptions occur. Secondly, we introduce an energy efficient, Context-Aware Cache Coding scheme (CACC) that adapts to network conditions and deployed applications. Using simple metrics, such as link loss rate and file size, CACC identifies the context in which Cache Coding is needed and then enforces it. Both emulation and real-world deployment on Android-based smartphones show that CACC improves the file delivery rate while reducing power consumption.

The rest of this paper is organized as follows. Section 2 briefly introduces the ICN architecture used for our implementation. Section 3 provides an overview of the foundations of network coding. Section 4 describes CACC, our implementation of Cache Coding in a delay-tolerant MANET. Section 5 evaluates the system with real-world scenarios, and demonstrates the benefits of energy-efficient CACC. Related work is discussed in Section 6. We conclude in Section 7.

## 2. ICN ARCHITECTURE

In ICN, the fundamental network primitive transitions from host-based addressing to content-based addressing (see examples [14, 3, 12]). Content is fetched by names, which allows intermediate nodes to act as either full or partial caches. Store-and-forward content dissemination enables a requestor the ability to fetch content pieces from multiple sources. This allows retrieval of content from nearby caches without reaching all the way back to the original content source, which produces improved throughput especially in severely disruptive scenarios. Cache Coding [16] can further increase the availability of (especially large) content in disruptive network environments by splitting the content into a number of redundantly coded elements that can be reassembled if a sufficient number of these coded elements are received. As we show in this paper, context information can be fed into Cache Coding to control the overhead.

We implemented and evaluated our context-aware cache coding (CACC) scheme in an ICN architecture called Information-CEntric Mobile Ad hoc Networking (ICEMAN) which was developed for the DARPA Content-Based Mobile Edge Networking (CBMEN) [4, 26, 27, 15]. ICEMAN is a modular, open-source, content-based mobile networking framework with a simple but expressive attribute-based mechanism for describing content and expressing interest, and in-network resolution to offer content dissemination by matching content to interest. Further details can be found in [5].

It should be noted that in an ICN architecture the decision to cache or not to cache at intermediate nodes can also be made context-aware. If an intermediate node is already a receiver, i.e., a requester, then the cache is available for free. However, if the intermediate note did not issue an interest for this file, then caching is an extra cost. If the link loss from the intermediate node to destination is high (e.g., due to enemy jamming), then caching is appropriate since this gives us more diversity and eventually more throughput. This way context awareness is used to optimize the trade off between energy (i.e., processing cost) and performance in the case of Cache Coding, initial coding, or no coding. However, in this paper paper, we only focus on using context awareness to determine when to enable Cache Coding.

## 3. CACHE CODING BASICS

Given the limited contact duration in MANET scenarios and the broadcast nature of wireless systems, the relays are most likely to obtain only partial files. The pieces (blocks) can be different from relay to relay, but some are replicated in several relays. When requesting a data object from multiple relays, the pieces are likely to be duplicate and arrive out of order. There will be gaps and missing pieces, which will make reliable reconstruction of the file difficult for a receiver. This is called the coupon collector problem (or last coupon problem). Network coding [13, 10] has been used in MANET [19] for data dissemination to overcome the last coupon problem. First, we briefly describe network coding. Then, we describe Cache Coding [16], our form of network coding that provides full protection from pollution attacks.

The algorithm of network coding is as follows. A source node publishes a file $F$. In order to disseminate the file in pieces using network coding, the source node first transforms $F$ into a set of $m$ vectors (i.e., chunks) $\mathbf{v_1}, ..., \mathbf{v_m}$ in an n-dimensional vector space over a finite field $\mathbb{GF}(2^8)$. These vectors are linearly combined by drawing, from the finite field $\mathbb{GF}(2^8)$, an encoding coefficient $\mathbf{e_i}$ to linearly combine with the vector to create $m$ coded blocks $\mathbf{b_1}, ..., \mathbf{b_m}$. The set of these coefficients then forms the encoding vector $\mathbf{e}$ with $[\mathbf{e_1}, ..., \mathbf{e_n}]$. To reconstruct the file, a node must receive enough linearly independent coded blocks to be able to perform matrix inversion. First, we take the transpose of the received vectors such that: $\mathbf{E^T} = [\mathbf{e^T}_\mathbf{1}, ..., \mathbf{e^T}_\mathbf{n}]$, $\mathbf{B^T} = [\mathbf{b^T}_\mathbf{1}, ..., \mathbf{b^T}_\mathbf{n}]$, and $\mathbf{V^T} = [\mathbf{v^T}_\mathbf{1}, ..., \mathbf{v^T}_\mathbf{n}]$. Then we take $\mathbf{E}^{-1}\mathbf{B}$ which will reconstruct all the original blocks in the file.

Note that the major advantage of network coding out-of-order blocks are no longer an issue. The requestor can retrieve coded blocks from any node, and reassemble the original file as long as it obtains a sufficient number of linearly independent blocks. Each independently generated coded blocks is equally innovative and useful to relays. The bandwidth saved by network coding due to control overhead and redundant data blocks at relays can be huge in a disruptive MANET ICN.

However, network coding is easily attacked. The primary advantage of Cache Coding [16] over network coding is that Cache Coding protects against pollution attacks while still achieving high throughput for file delivery. In Cache Coding, only sources and intermediate nodes have fully reconstructed the file, and are able to encode and propagate the coded blocks into the network.

Cache Coding provides good diversity, which is necessary for network coding efficiency, without requiring unrestricted network coding at all intermediate nodes. If we recode at intermediate nodes (without Cache Coding), we must protect the data from pollution attacks, e.g., by using homomorphic codes. With Cache Coding, a node signs the cache before mixing packets. When the generation is decoded, if it fails the signature, it is discarded. Because signed blocks cannot be repudiated, malicious intermediate nodes are easily detected.

## 4. ENERGY-EFFICIENT CONTEXT-AWARE CACHE CODING

While Cache Coding has the advantages as previously described, it requires extra bandwidth to carry the coefficients and handle the computational resources dedicated to the encoding/decoding processes. The tradeoffs of performance gain and overhead for disruptive MANET ICNs must be managed by our proposed energy-efficient CACC.

Our proposed contribution CACC aims to adaptively switch between Cache Coding, low-overhead fragmentation, and atomic transmission of data objects. Depending on the context, CACC automatically turns Cache Coding on and off for a given data object. The objective is to eliminate unnecessary bandwidth consumption due to increased header size and processing overhead when Cache Coding is unlikely to improve end-to-end performance, but be ready to trigger Cache Coding instantly in emergencies by using context indicators.

## 4.1 Context Indicators

The nature of Cache Coding methods requires fragmenting the data object being transmitted to utilize the benefit of transmitting random coded blocks, either when frequent retransmissions are required from one source or when multiple content sources may be leveraged. However, in the case when a data object is small, fragmenting that object may be less efficient than simply requesting an atomic re-transmission due to the fact that identification of the fragments requires additional overhead. Moreover, Cache Coding incurs even higher overhead due to the required coefficients in each coded block. Therefore, Cache Coding should remain disabled in the case when the file size is relatively small compared to the basic fragment unit.

The major advantage of Cache Coding is its resilience to intermittent links due to high mobility. This advantage comes from the fact that Cache Coding is randomized and all coded blocks contain equal entropy. When the link breaks down often, it is difficult to predict which fragmented blocks get lost without proper feedback. With Cache Coding, however, all sources/caches can send innovative blocks (i.e., each one of them is linearly independent with the rest of the received fragments) without waiting for feedbacks, which leads to a higher chance of finishing full file transmission given a reasonably short period of time over one hop. Another important factor to consider is the coefficient overhead of Cache Coding. The coefficient overhead increases the transmission unit size and may worsen the performance. Therefore, it is expected that Cache Coding should be disabled when the link is not lossy.

Based on the above observations, we propose two context indicators.

- **Application-related context indicator**: The *data object size* needs to be considered when deciding whether the data object is more suitable for Cache Coding (i.e., the overhead can be compensated) vs. atomic (re-)transmission.

- **Network condition-related context indicator**: When the connectivity is intermittent and the contact time is short, the *link loss rate* can reflect the current situation (i.e., network condition and node mobility) to determine whether Cache Coding can speed up the data delivery from multiple sources/caches.

## 4.2 CACC

Our adaptive Cache Coding algorithm takes the following steps. By default, Cache Coding is disabled for new data objects and destination nodes.

1. CACC monitors the link loss rate for all known one-hop neighbors. The loss rate estimation is simply a weighted moving average over a fixed interval of time. The link loss rate is estimated based on the number of successful receptions of periodic beacons at the link layer. The beacon interval $\triangle t$ is parameterized and can be adjusted to vary sensitivity. For downstream nodes, if the downstream link quality is good, fragmentation can be used for subsequent hops after receiving Cache Coded blocks and reconstructing content. We use the following equations to compute the loss rate estimate:

$$\alpha = 1 - e^{\frac{\triangle t}{-W}} \tag{1}$$

$$l_n(t) = \alpha \cdot b_i + (1-\alpha) \cdot l_n(t-1) \tag{2}$$

   where $W$ is defined as the length of the interval of time to measure the loss estimate over, $\triangle t$ is defined as the duration between beacons, and $b_i$ is 1 if no beacon was received (indicating a loss) and 0 if a beacon was received (indicating no loss). $l_n(0)$ is defined to be 1, indicating that we perform cache coding until the loss rate settles down to a point where we can safely stop using it.

2. At the sender side (i.e., the sender can be a cache or a data source), Cache Coding is enabled for data objects that satisfy the following two conditions:

$$l_n(t) > l_{th} \tag{3}$$

$$s(d) > s_{th} \tag{4}$$

   where $l_{th}$ and $s_{th}$ are the threshold values of link loss rate and data object size, respectively. $l_n(t)$ is the average link loss rate at time $t$ for the 1-hop neighbor node $n$. $s(d)$ is the size of data object $d$.

   Note that even relay nodes that are not data sources nor caches (i.e., did not issue an interest for this file) must apply the CACC procedure. More precisely, if the link loss criterion determines that Cache Coding should be used, the Relay must start accumulating blocks of the file in question until it has cached the entire file. At that point it switches to Cache Coding. It is intuitive to understand why arbitrary relays (not interested in the file) must cache code. In fact, a relay node may be attacked by an adversarial jammer and drop all the packets. Recovery is possible only if the Relay nodes on the cut-set under attack enter the Cache Code mode, thus providing enough diversity to overcome the attack.

3. At the receiver side (i.e., the receiver may be a relay node or the intended receiver), if a Cache Coded block is received from any of its neighbors, the previously received un-encoded fragments are converted to encoded blocks.

The conversion from an un-encoded fragment to an encoded block allows CACC to recycle already received fragments rather than waiting sufficient numbers of innovative blocks are generated from a source and disseminated to a receiver. In CACC, the $i$-th fragment of a file can be seen as a special case of coded block in which the coefficients used to encode this block is a unit vector in which the $i$-th element is 1 and all other elements are 0. Therefore, the fragment can be converted to a coded block by simply encoding it with such a unit vector. For example, to convert the second fragment $f_2$ of a 4-fragment file, we compute the converted coded block by $[0, 1, 0, 0][\mathbf{0}, \boldsymbol{f}_2, \mathbf{0}, \mathbf{0}]$. In this way, all received fragments and encoded blocks can be used for decoding the original data object so that all successful transmissions are utilized.

## 4.3 Implementation

The ICEMAN architecture [26, 27] is event-driven, modular, and layer-less, which provides flexibility and scalability. Central in the architecture is the kernel. It implements an event queue, over which managers that implement the functional logic communicate. Managers are responsible for specific tasks such as managing communication interfaces, encapsulating a set of protocols, and forwarding content.

In ICEMAN, files are opportunistically cached at mobile nodes to favor future file requests. Files can be downloaded in parallel from multiple caches to make downloads reliable and fast. Cache Coding across parallel caches further improves the throughput. However, in intermittent connectivity, caches may often be partial. Thus, these caches cannot be signed since the signature implies that the intermediate node has received the full file, has verified the signature and has replaced in each block the originator signature with its own.

The traditional unrestricted network coding in which all relays may mix (i.e., re-encode) any available chunks even if the cached data object is partial, exposes security concerns in an ICN due to the possibility of pollution attack. Therefore, we apply the *cache coding*, which allows only the caches who have the full data objects to reencode, generate, and sign new coded blocks. Caches holding only partial data objects are only allowed to forward the coded blocks they have received as is. In this way, attackers can be identified and blacklisted by looking at the signature.

In ICEMAN, interests are composed of key-value pairs to describe the content the requestor is searching for. Suppose node $n$ broadcasts an interest and gets metadata notification from the source node $s$ as a match to its interest. Node $n$ then starts downloading the blocks from node $s$. During this process, node $n$ learns the unique content identifier ($Cid$) of the file carried in each block, which can be used to optimize the content flow of blocks.

For this purpose, we introduce the concept of *precise* interests when the requestor learns, from its neighbors, $Cid$ for which it is searching for. Using $Cid$, node $n$ can check the Bloom filters received from other neighbors and determine if there are other caches for file $f$. Knowing that other neighbors have a copy of $f$, $n$ can send them *precise* interests, which is essentially $Cid$ of $f$ and can start pulling the blocks from multiple caches in parallel. This multiple cache downloading via precise interests improves the performance of Cache Coding. In this case, Bloom filters (cache summaries that disseminated side by side with interests to provide a view of the files available, both complete and incomplete files) can serve as routing tables to indicate the direction to the cache.

CACC switches between *cache coding* and low-overhead fragmentation by generating an *event* in ICEMAN architecture. The switching occurs at a sender for specific target nodes; that is, the decision is based on each pairwise link. The link loss rates are calculated as a moving average of the percentage of lost beacon packets over given period of time. To measure the packet loss, an event is sent from the Ethernet connectivity module of ICEMAN to notify the loss estimation module upon successfully receiving a beacon. The loss estimation module then decides to generate another event to CACC that in turn triggers switching between Cache Coding and fragmentation. A node switches to Cache Coding for data transmissions targeting nodes associated with the high loss links, and switches back to fragmentation on other links upon receiving this event. The implementation of CACC parameterizes $l_{th}$ and $s_{th}$, via user-defined configuration files.

## 5. EVALUATION

We study the scenarios where Cache Coding has the highest throughput gain and the scenarios where it is not strictly required. We use these experimental results to guide intuition about the contexts in which Cache Coding should and should not be utilized.

## 5.1 Setup

In order to evaluate the performance on larger mobile scenarios, our test framework emulates mobile network scenarios on a Linux server. The network emulation is done using EMANE 802.11 [2]. EMANE includes a path loss model and interfaces with the Common Open Resource Emulator, CORE 4.3 [6]. We use EMANE and CORE together to run realistic emulations of mobile performance on a Linux server using the same codebase that runs on Android devices with real testbed experiments. EMANE models the network topology, and CORE is responsible for the mobility model and moving nodes around in accordance with scenario requirements. We use CORE for compartmentalization so that multiple virtual ICEMAN nodes can exist on the same physical machine; through the use of resource and network isolation.

For Android testing, we ran tests on Nexus S smartphones running Gingerbread. In order to emulate the resource constrained nature of an Android device, we use the cpulimit [1] tool to limit the CPU time allocated to ICEMAN processes.

To capture the energy requirement, we use a normalized energy utility measurement. We define this as measuring the total number of files delivered and the total CPU time needed. We then normalize both file delivery and CPU overhead to fragmentation (no coding). Finally, we divide the normalized number of files delivered by the normalized CPU overhead. It has been shown previously that coding induces noticeable computation overhead on mobile devices [25, 24]. Thus, we capture the CPU overhead as an distinguishing energy indicator when comparing coding versus no coding.

## 5.2 Parameters

Our test framework runs publisher and subscriber applications on each ICEMAN node that communicates with the ICEMAN process on each virtual node according to the requirements set by the scenario. Each publisher and subscriber application keeps logs of the data objects that are sent and received. These logs are then analyzed, along with emulation output from EMANE and CORE, to study the bandwidth, latency, and delivery rates from the experiment.

We configured our system to use Cache Coding for files larger than 32KB. The block size for Cache Coding operations was also set to 32KB. The window parameter $W$ for the loss estimate calculation was set to 30 seconds. Data Objects were disseminated using UDP broadcast. The UDP broadcast mode precludes MAC layer ACKs so there was no loss detection and retransmission. Through out our experiments, we use *ieee802.11abg* link provided by EMANE with an omnidirectional antenna gain of -5 dbi and a system noise factor of 4 db in freespace loss model.

## 5.3 Micro Benchmark

We first evaluate the performance overhead of Cache Coding versus fragmentation in a simple 2-node scenario. We run a 6 minute scenario in which two nodes publish 60 objects each, in order to measure the overhead of NetCode versus Fragmentation under ideal conditions on a resource constrained device.

In the Micro Benchmark, there is good connectivity between the two Android phones, and Frag performs better. In Table 1, NetCode imposes extra overhead due to the expensive coding operations, while Frag allows data to be transmitted quickly and with lower power usage. This motivates the development of CACC, which switches intelligently between them.

| | Files Delivered | Normalized Energy Utility |
|---|---|---|
| Frag | 30 | 1.0 |
| NetCode | 20 | 0.33 |

Table 1: Micro Benchmark: In static scenarios with good connectivity, NetCode has unncessary overhead.

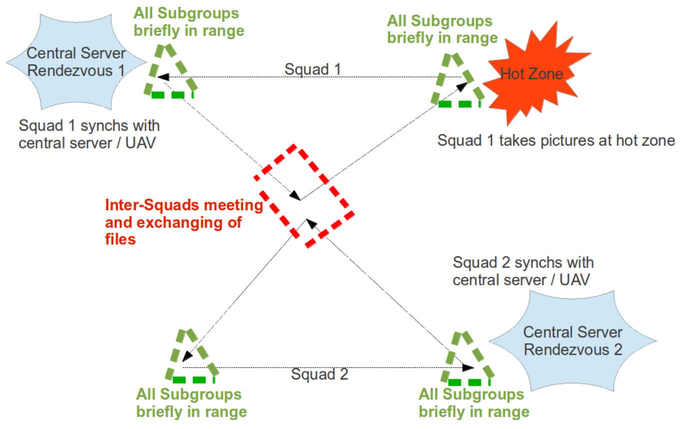
All Subgroups briefly in range
Central Server Rendezvous 1
Squad 1
All Subgroups briefly in range
Hot Zone
Squad 1 synchs with central server / UAV
Squad 1 takes pictures at hot zone
Inter-Squads meeting and exchanging of files
Squad 2 synchs with central server / UAV
Central Server Rendezvous 2
Squad 2
All Subgroups briefly in range
All Subgroups briefly in range

Figure 1: Search patrol scenario with 30 nodes: 2 squads (composed of 3 sub-squads) walk in a triangle pattern between 3 rendezvous points.

## 5.4 Scenarios

Our evaluation tests CACC across two scenarios: a search patrol scenario in Section 5.4.1, and a data mule scenario in Section 5.4.2; to illustrate the behaviour of CACC in dynamic settings.

### 5.4.1 *Search Patrol Scenario*

Consider the search patrol model in Figure 1, where a search patrol visits rendezvous points and shares reconnaissance and updates information. In this search patrol scenario, there are 2 squads with 14 members each. Each squad is composed of 2 subgroups. Intra-squad communication occurs periodically at each rendezvous point (i.e., central servers, hot spot, and red box in Figure 1).

Both the central servers publish two 512KB files each. Each scout in each subgroup takes a 1MB picture at the hot zone and performs intra-subgroup sharing. This results in a total of 60 files published and shared to a total of 30 subscribers. Intra-squad sharing occurs only at each rendezvous point. Inter-squad file sharing occurs in the middle ground as illustrated as the red box in Figure 1. Due to intermittent connectivity, each node only has partial caches. The partial caches consist of the files from the central servers, and pictures taken at the hot zone. If there are partial files, intra-squad sharing

| | Files Delivered | Normalized Energy Utility |
|---|---|---|
| Frag | 158 | 1.0 |
| CACC(0.2) | 513 | 1.08 |
| CACC(0.4) | 489 | 0.92 |
| CACC(0.6) | 363 | 0.87 |
| CACC(0.8) | 382 | 0.71 |
| NetCode | 396 | 0.71 |

Table 2: Search Patrol Scenario: CACC is able to utilize both NetCode and Frag for improved delivery rates and reduced power consumption.

continues as each squad walks to next rendezvous point. This scenario continues for 720 seconds.

### 5.4.2 Data Mule Scenario

In the data mule model, two large squads (4x4 and 3x4 members each) are connected by a few data mules which ferry files. Nodes in a squad are 30 meters apart. Intra-squad communication is conducted by multi-hop connections while inter-squad communication is facilitated by three data mules. In this scenario, every node in each squads publishes two 1MB files, one every five minutes. The three data mules publish five 512K files, one every minute. All nodes subscribe to all content. The data mule speed is 1.4 m/s. Each data mule is out of range of other data mules and communicates with each squad for 60 seconds. The duration of this scenario is 600 seconds. This model allows us to evaluate the effects of large, internally connected groups that are interconnected with each other by a low-bandwidth connection. We expect to see the coupon collector problem taking effect and slowing down the file reconstruction between the two groups.

## 5.5 Results

To measure the effectiveness of dissemination, we measure both the delivery rate, in terms of the number of data objects delivered, as well as the normalized energy utility as defined in Section 5.1.

We compare across three different schemes, low overhead fragmentation (Frag); cache coding without any context aware switching (NetCode), and network coding with context aware switching (CACC). We use the notation CACC($l_{th}$) to indicate the usage of a CACC with the given value of $l_{th}$.

We experiment across various values of $l_{th}$, using values of 20%, 40%, 60%, and 80%. Conceptually, Frag is equivalent to using CACC with an $l_{th}$ value greater than 100%; as $l_{th}$ can never exceed 1, network coding will never be enabled. NetCode is equivalent to using an $l_{th}$ value of 0% as network coding will then immediately be enabled.

### 5.5.1 Search Patrol Scenario

In this scenario, the subsquads move as groups and

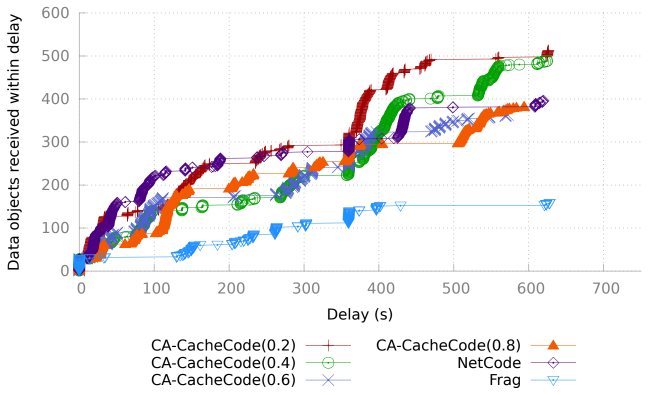

Figure 2: Search Patrol Scenario: CACC delivers more data objects than NetCode using less power.

the channel loss rates between intra-squad links are low, while the average loss rates of inter-squad links are relatively high. Additionally, the rendezvous time is short so that data objects are delivered in parts. Therefore, NetCode has a clear advantage for transmissions between nodes from different squads. However, since the number of nodes in the mobile group is small and the number of data objects to be exchanged within the mobile group is few, the interference caused by channel contention is relatively low. Therefore, NetCode is not required for intra-subsquad communications and would just introduce unnecessary overhead. Frag gets the fewest data objects through, since it gets hit with the coupon collector problem.

CACC combines the best of both worlds, it is able to utilize low-overhead Fragmentation for intra-subsquad transmissions and use NetCode when needed for inter-subsquad transmissions. From the results we can see that CACC uses less power than NetCode; and delivers many more data objects in the case of CACC(0.2) and CACC(0.4).

Figure 2 and Table 2 illustrate this point clearly. Note that the delivery rate is a function of the mobility, and jumps when the subsquads interact. Initially, NetCode performs the best and has the lowest latency, at the expense of power consumption; NetCode always uses expensive network coding operations while CACC tries to save power and use low-overhead Fragmentation. Over time, when enough blocks get through and the squads meet again, CACC is able to reconstruct data objects and improve delivery rates.

### 5.5.2 Data Mule Scenario

The Data Mule scenario illustrates the benefits of using CACC for delivery rates versus simply using NetCode or Frag. Intra-squad sharing is reliable, but inter-squad sharing depends on how efficiently blocks can be transmitted by the data mules to the other squad. In this scenario, the delivery rates are a function of the mobility. We expect that all intra-squad subscriptions

| | Files Delivered | Normalized Energy Utility |
|---|---|---|
| Frag | 114 | 1.0 |
| CACC(0.2) | 256 | 1.29 |
| CACC(0.4) | 202 | 1.21 |
| CACC(0.6) | 287 | 1.45 |
| CACC(0.8) | 262 | 1.31 |
| NetCode | 248 | 1.25 |

Table 3: Data Mule Scenario: CACC delivers more data than NetCode while using similar amounts of power.

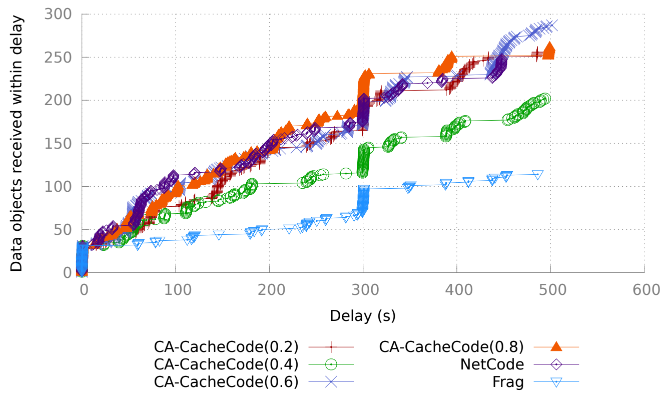

Figure 3: Data Mule Scenario: CACC delivers more data than NetCode and Frag.

will complete successfully, while the delivery rate for inter-subsquad subscriptions will depend on the mules. If the mules have a shorter stay, the spikes in data object delivery will be closer together. However, the spikes would be shorter as the mules would not be able to receive/send enough data objects when they are in contact with the squads. A stay duration of 60 seconds, as in our scenario, is a reasonable middle ground.

In Table 3, we can see CACC outperforms both NetCode and Frag. CACC(0.6) delivers more data objects when compared to plain NetCode. Other cases (except CACC(0.4)) perform like NetCode in terms of delivery rate. CACC is able to utilize the network context to learn that blocks should be sent to the mules. It thus pays the cost of network coding for those nodes, but avoids expensive network coding operations when sharing data over reliable intra-squad links. We note that the power consumption is similar to NetCode, but more data is delivered.

Figure 3 illustrates this point. Frag takes much longer to deliver data objects intra-squad initially, due to the coupon collector problem. Very few data objects are delivered through the mules, again due to the same problem. NetCode, since it does not suffer from the coupon collector problem, is able to deliver more data objects, and deliver them quickly. Initially, the performance of CACC is somewhere in between NetCode and Frag. This is due to the fact that at startup a lot of data is being exchanged, and CACC needs to detect the network state. CACC detects the initial lossy channel and switches to network coding. Later on, past the 100 second mark, when enough blocks and fragments come in through the mules, intra-squad sharing can proceed through low-overhead fragmentation and thus delivery rates are improved.

| | Files Delivered | Normalized Energy Utility |
|---|---|---|
| Frag | 60 | 1.0 |
| CACC(0.2) | 330 | 7.93 |
| CACC(0.4) | 353 | 7.06 |
| CACC(0.6) | 234 | 3.58 |
| CACC(0.8) | 178 | 2.73 |
| NetCode | 252 | 5.81 |

Table 4: Android Search Patrol Scenario: CACC(0.2) delivers the most data while using the least power.

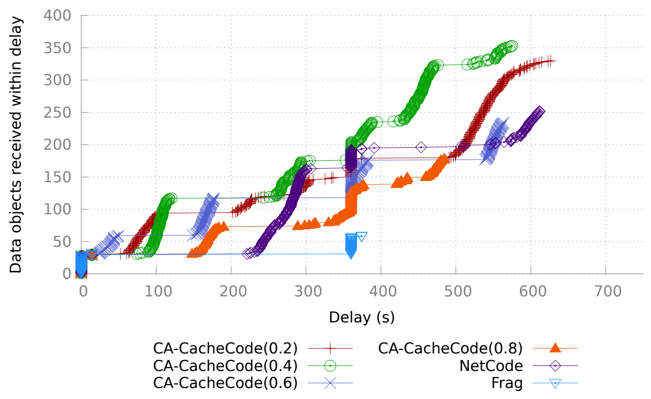

Figure 4: Android Search Patrol Scenario: CACC delivers more data than NetCode and Frag.

### 5.5.3 *Search Patrol Scenario on Android*

Results for the Search Patrol scenario on Android are presented in Table 4 and Figure 4. We can see that NetCode initially takes longer to deliver data objects, due to the time required to encode and decode blocks on a slower CPU. In this time, CACC, using a mix of fragmentation and network coding, is able to deliver data objects with lower latency. Note that Frag is unable to deliver many data objects as it is hit hard by the coupon collector problem.

In terms of power usage, we see that CACC(0.2) uses the least power, while other CacheCode settings use more power. In addition, Frag uses more power per data object than NetCode in this scenario, due to the fact that delivery rates are low. This illustrates that the choice to use low-overhead transmissions must also consider the fact that data delivery may be adversely affected. We can also observe that CACC is adaptive and able to both save power and increase delivery rates.

## 6. RELATED WORK

We summarize a few representative network coding approaches and their implementations that also propose context-awareness, even in some cases within the scope of ICNs, and highlight the differences from our study.

### 6.1 Network coding in ICNs

Montpetit et al. [23] have identified network coding within content-centric networking (CCN) as a strategy with tremendous potential. However, their work only presented an architecture, rather than reporting on an empirical analysis. Further, we have introduced the idea of adaptively enabling network coding, explained the need for context-aware network coding to preserve energy resources on resource constrained devices, and implemented our ideas on Android devices. We have also emulated the system, the network-wide effects, and reported our findings.

Wu et al. [28] have implemented and evaluated the benefits that network coding provides for cache hit rates in content centric networks. Interestingly, the evaluation used unrestricted coding applied to real traces from PPTV (peer-to-peer video streaming). However, their work does not address selectively enabling network coding, which may not always be required in a wired peer-to-peer system. Moreover, it does not address pollution attack protection in unrestricted coding. Our solution detects when to enable network coding, a characteristic that is of critical importance in mobile environments with limited power.

### 6.2 Context-aware network coding

Previous work on network coding based on context has mostly been associated with forwarding in disrupted networks. It has focused on adjusting the degree of redundancy in the encoding. MORE [7] is the earliest work that proposes adaptive network coding for such environments. MORE relays opportunistically form multiple paths through which packets are re-encoded and forwarded. CodeMP [8] further studies adaptive network coding based on measured loss rates that affect TCP sessions. While this family of adaptive network coding approaches adjust the use of multiple paths and degree of redundancy using link loss rate estimates, they focus on connected networks with unstable channels. In contrast, we apply network coding in delay-tolerant networking scenarios. We have also conducted our experiments with more realistic channel models.

Existing work on using network coding in delay-tolerant networks (DTNs [22]) focuses on reducing the number of transmissions needed for epidemic routing or probabilistic forwarding. Lin et al. [20] studied the tradeoff between performance and resource consumption in DTNs. They propose spreading more coded packets than are needed to reduce the number of transmissions required. Chuah et al. [9] proposed CANCO, which only spreads coded packets among some of the nodes encountered. Delivery predictability and friendliness are used as metrics to decide which nodes to use.

Our work differs from previous work in several respects. First, our goal is to maximize the delivery rate while reducing the energy consumption by selectively enabling and disabling network coding. Prior work is unable to adapt to improved network conditions. In such cases, always-on network coding imposes a cost, and depletes precious power on resource constrained devices. Second, the context we use is the condition of the network, rather than the history of encounters or social relationships with other nodes. Third, we employ network coding by broadcasting blocks over UDP connections to neighbors, rather than reducing transmissions by forwarding to only a subset of nodes. More importantly, we provide an algorithm and evaluate an implementation that automatically switches between fragmenting and network coding the content, based on the runtime context. Of equal significance, we have performed emulations of real-world scenarios and implemented the system on Android devices. In contrast, earlier work on context-aware and adaptive network coding is limited to theoretical analysis or simulations of limited scenarios. It is worth noting that CACC can be used with previously proposed coding-aware routing protocols, such as [20] and [9].

## 7. CONCLUSION

In this paper, we have evaluated the performance of a cache oriented, network coding enabled ICN in two different versions: with and without the use of Cache Coding. More precisely, we have researched the "context" for activating/deactivating Cache Coding. We call this adaptive strategy Context-Aware Cache Coding (CACC).

We demonstrated that while Cache Coding is necessary in environments with intermittent connectivity in order to reliably disseminate files, in "connected topology" conditions the magnitude of the throughput increase may not warrant the higher overhead of Cache Coding. In such situations, CACC detects and disables Cache Coding. By testing CACC on Linux and Android, we showed that opportunistic disabling of Cache Coding reduces the latency and power consumption, and maintains or even increases the data delivery rate.

In this study we mainly focused on sender-driven switching, where a sender continuously monitors the loss threshold in order to decide when to switch modes. In future work we plan to investigate a receiver-driven strategy. In the latter, the receiver will monitor its neighbors to detect if they are receiving coded blocks. If enough neighbors are receiving coded blocks, the receiver may request the sender to switch to Cache Coding in order for the receiver to take advantage of the diversity of nearby coded partial caches. Additionally, we plan to explore optimal selection of the "context" parameters.

## 8. ACKNOWLEDGEMENTS

This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) and SPAWAR Systems Center Pacific (SSC Pacific) under Contract N66001-12-C-4051 and by SRI International. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. Approved for Public Release, Distribution Unlimited, Case 23042.

## 9. REFERENCES

[1] cpulimit. https://github.com/opsengine/cpulimit.

[2] Emane. http://cs.itd.nrl.navy.mil/work/emane/.

[3] Named-Data Networking (NDN). http://named-data.net.

[4] Content-based mobile edge networking. http://www.darpa.mil/Our_Work/STO/Programs/Content-Based_Mobile_Edge_Networking_(CBMEN).aspx, 2013.

[5] Encoders (Edge Networking with Content-Oriented Declarative Enhanced Routing and Storage). http://encoders.csl.sri.com/, 2015.

[6] J. Ahrenholz, C. Danilov, T. Henderson, and J. Kim. CORE: A real-time network emulator. In *Military Communications Conference, 2008. MILCOM 2008. IEEE*, pages 1–7, 2008.

[7] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In J. Murai and K. Cho, editors, *Proceedings of the ACM SIGCOMM 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Kyoto, Japan, August 27-31, 2007*, pages 169–180. ACM, 2007.

[8] C.-C. Chen, G. Tahasildar, Y.-T. Yu, J.-S. Park, M. Gerla, and M. Sanadidi. Codemp: Network coded multipath to support tcp in disruptive manets. In *Mobile Adhoc and Sensor Systems (MASS), 2012 IEEE 9th International Conference on*, pages 209–217. IEEE, 2012.

[9] M. Chuah, P. Yang, S. Roy, and B. Sheng. Performance evaluation of dissemination schemes for coded packets in heterogeneous sparse ad hoc networks. *Ad Hoc And Sensor Wireless Networks*, 15(2-4):151–181, 2012.

[10] C. Fragouli, J. L. Boudec, and J. Widmer. Network coding: an instant primer. *Computer Communication Review*, 36(1):63–68, 2006.

[11] C. Fragouli, J. Widmer, and J.-Y. Le Boudec. A network coding approach to energy efficient broadcasting: From theory to practice. In *INFOCOM 2006*, pages 1–11, April 2006.

[12] Haggle: An innovative Paradigm for Autonomic Opportunistic Communication Research project. http://www.haggleproject.org/.

[13] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *Information Theory, IEEE Transactions on*, 52(10):4413–4430, 2006.

[14] V. Joacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *CoNEXT '09: Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, pages 1–12. ACM, 2009.

[15] J. Joy, Y. Yu, M. Gerla, S. Wood, J. Mathewson, and M. Stehr. Network coding for content-based intermittently connected emergency networks. In S. Helal, R. Chandra, and R. Kravets, editors, *The 19th Annual International Conference on Mobile Computing and Networking, MobiCom'13, Miami, FL, USA, September 30 - October 04, 2013*, pages 123–126. ACM, 2013.

[16] J. Joy, Y. Yu, V. Perez, D. Lu, and M. Gerla. A new approach to coding in content-based manets. *JCM*, 9(8):588–596, 2014.

[17] A. Keshavarz-Haddadt and R. Riedi. Bounds on the benefit of network coding: Throughput and energy saving in wireless networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages –, April 2008.

[18] S.-H. Lee, M. Gerla, H. Krawczyk, K.-W. Lee, and E. Quaglia. Performance evaluation of secure network coding using homomorphic signature. In *Network Coding (NetCod), 2011 International Symposium on*, pages 1–6, 2011.

[19] U. Lee, J.-S. Park, J. Yeh, G. Pau, and M. Gerla. Code torrent: content distribution using network coding in vanet. MobiShare '06, pages 1–5, New York, NY, USA, 2006. ACM.

[20] Y. Lin, B. Li, and B. Liang. Efficient network coded data transmissions in disruption tolerant networks. In *INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 13-18 April 2008, Phoenix, AZ, USA*, pages 1508–1516. IEEE, 2008.

[21] D. S. Lun, M. MÃľdard, T. Ho, and R. Koetter. Network coding with a cost criterion. In *in Proc. 2004 International Symposium on Information Theory and its Applications (ISITA 2004*, pages 1232–1237, 2004.

[22] A. McMahon and S. Farrell. Delay- and disruption-tolerant networking. *IEEE Internet Computing*, 13(6):82–87, 2009.

[23] M.-J. Montpetit, C. Westphal, and D. Trossen. Network coding meets information-centric networking: an architectural case for information dispersion through native network coding. In *Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking*

*Design - Architecture, Algorithms, and Applications*, NoM '12, pages 31–36. ACM, 2012.

[24] H. Shojania and B. Li. Random network coding on the iphone: fact or fiction? In W. T. Ooi and D. Xu, editors, *Network and Operating System Support for Digital Audio and Video, 19th International Workshop, NOSSDAV 2009, Williamsburg, VA, USA. June 3-5, 2009, Proceedings*, pages 37–42. ACM, 2009.

[25] H. Shojania and B. Li. Tenor: making coding practical from servers to smartphones. In A. D. Bimbo, S. Chang, and A. W. M. Smeulders, editors, *Proceedings of the 18th International Conference on Multimedia 2010, Firenze, Italy, October 25-29, 2010*, pages 45–54. ACM, 2010.

[26] S. Wood, J. Mathewson, J. Joy, M. Stehr, M. Kim, A. Gehani, M. Gerla, H. R. Sadjadpour, and J. J. Garcia-Luna-Aceves. ICEMAN: A practical architecture for situational awareness at the network edge. In N. Martí-Oliet, P. C. Ölveczky, and C. L. Talcott, editors, *Logic, Rewriting, and Concurrency - Essays dedicated to José Meseguer on the Occasion of His 65th Birthday*, volume 9200 of *Lecture Notes in Computer Science*, pages 617–631. Springer, 2015.

[27] S. Wood, J. Mathewson, J. Joy, M.-O. Stehr, M. Kim, A. Gehani, M. Gerla, H. Sadjadpour, and J. Garcia-Luna-Aceves. ICEMAN: A system for efficient, robust and secure situational awareness at the network edge. In *Military Communications Conference, MILCOM 2013-2013 IEEE*, pages 1512–1517. IEEE, 2013.

[28] Q. Wu, Z. Li, and G. Xie. Codingcache: multipath-aware ccn cache with network coding. In *Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking*, ICN '13, pages 41–42. ACM, 2013.

[29] Y. Wu, P. Chou, and S.-Y. Kung. Minimum-energy multicast in mobile ad hoc networks using network coding. *Communications, IEEE Transactions on*, 53(11):1906–1918, Nov 2005.